2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

Locally Constrained Decision Making via Two-Stage Distributed Simplex

Mathias Bürger, Giuseppe Notarstefano, and Frank Allgöwer

Abstract— **In this paper we propose a distributed algorithm for solving linear programs with combinations of local and global constraints in a multi-agent setup. A fully distributed and asynchronous algorithm is proposed. The computation of the local decision makers involves the solution of two distinct (local) optimization problems, namely a local copy of a global linear program and a smaller problem used to generate "problem columns". We show that, when running the proposed algorithm, all decision makers agree on a common optimal solution, even if the original problem has several optimal solutions, or detect unboundedness and infeasibility if necessary.**

I. Introduction

A reliable decision making by distributed agents is a core requirement for all envisioned self-organizing, networked systems. Numerous applications, ranging from sensor networks to robotic systems, need the possibility to solve optimization and decision problems, while the problem information is distributed throughout the network.

Distributed optimization in multi-agent systems has recently attained significant attention in the literature, see e.g. [1], [2], [3] for general convex programming. Several contributions also deal explicitly with distributed linear programs, including [4], [5]. Of particular relevance to this work is the work of Notarstefano and Bullo [6], see also [7], which proposes the *constraints consensus* as distributed algorithm for solving abstract optimization problems (including dual linear programs). Building on this work, we have proposed in [8] the Distributed Simplex as a primal algorithm for solving distributed linear programs in primal form. The actual paper complements our previous work.

The contributions of this paper are as follows. We propose an asynchronous algorithm which solves distributed linear programs with a particular structure. We are concerned with problems where several agents supervise their own decision variables, which are required to be contained within (bounded or unbounded) polyhedra. The agents should find an agreement and adjust their decision variables such that the sum of their individual costs is minimized while a set of global equality constraints (involving decision variables of all agents) are satisfied. We propose an algorithm in which each agent has internally a two-stage structure. Each agent treats internally two optimization problems: a local version of the full problem and a sub-problem which is used to generate new problem columns. Our algorithm adapts here the *column generation* idea from the classical Danzig Wolfe decomposition. The different agents exchange asynchronously primal information concerning only their local copies of the original problem, while the sub-problems remain private. This gives rise to a novel fully distributed algorithm without coordination unit. The novel algorithm can be seen as a generalization of the Distributed Simplex algorithm, we have presented in [8]. We prove that our algorithm can detect infeasibility or unboundedness of the problem if necessary. If instead a finite optimal basis exists, we show that all agents will agree on a common optimal basis in finite time. We also point out that it can in general not be predicted, on which optimal solution the agents agree if the optimal solution is not unique.

The remainder of the paper is organized as follows. We present the problem setup in § II, including the original optimization problem and the multi-agent system. In § III the classical decomposition and column generation method is reviewed. The main result of this paper is presented in § IV, where we combine the Distributed Simplex algorithm with the column generation idea, to propose an asynchronous distributed algorithm. We prove the correctness of the new algorithm.

M. Bürger and F. Allgöwer thank the German Research Foundation (DFG) for financial support within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart and within the Priority Program 1305 "Control Theory of Digitally Networked Dynamical Systems". The research of G. Notarstefano has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 224428 (CHAT) and n. 231378 (CO3AUV) and from the Italian Minister under the national project "Sviluppo di nuovi metodi e algoritmi per l'identificazione, la stima bayesiana e il controllo adattativo e distribuito."

M. Bürger and F. Allgöwer are with the Institute for Systems Theory and Automatic Control, University of Stuttgart, Pfaffenwaldring 9, 70550 Stuttgart, Germany, {mathias.buerger, frank.allgower}@ist.uni-stuttgart.de.

Giuseppe Notarstefano is with the Department of Engineering, University of Lecce, Via per Monteroni, 73100 Lecce, Italy, giuseppe.notarstefano@unile.it.

II. Problem Statement

We study a multi-agent decision problem where distributed decision makers, equipped with individual costs and constraints, have to adjust their decisions in a way that is aligned with a coupling constraint, involving all the decision variables.

A. Distributed Linear Programs

We are given a group of N decision makers, in the following called agents. Each agent $i \in \{1, \dots, N\}$ supervises a vector of decision variables $x_i \in \mathbb{R}^{n_i}$. It aims to minimize an individual linear cost

$$\mathrm{c}_i^T x_i,$$

$\mathrm{c}_i \in \mathbb{R}^{n_i}$, while keeping the decision variable within the non empty *polyhedron*

$$x_i \in \mathcal{X}_i.$$

Note that the constraint sets $\mathcal{X}_i$ are not necessarily bounded. In particular they might be restricted in some directions, while they are unbounded in other directions. We require the agents, furthermore, to adjust their decision variables x_i in a way that satisfies the *coupling constraints*

$$\sum_{i=1}^{N} \mathrm{A}_i x_i = \mathrm{b}_0,$$

where $\mathrm{b}_0 \in \mathbb{R}^d$ and $\mathrm{A}_i \in \mathbb{R}^{d \times n_i}$. While satisfying the coupling constraints the agents should minimize the global cost, which we interpret simply as the sum of all individual cost functions. We therefore consider in this study the *distributed linear program*

$$OP: \qquad \phi := \min_{x_i \in \mathcal{X}_i} \sum_{i=1}^{N} \mathrm{c}_i^T x_i \tag{1}$$

$$\text{s.t.} \sum_{i=1}^{N} \mathrm{A}_i x_i = \mathrm{b}_0. \tag{2}$$

This formulation of a distributed linear program extends the formulation we studied in our previous work [8]. We allow now explicitly for the individual constraints $x_i \in \mathcal{X}_i$, which might

eventually consist of a large number of additional equality or inequality constraints. In the following we will develop an algorithm to solve problems of the form OP within multi-agent systems. The proposed algorithm, relying on the Distributed Simplex algorithm we presented in [8], handles the constraints $x_i \in \mathcal{X}_i$ in a "local" manner. The computation and communication model of the multi-agent system is introduced in the next section.

B. Communication Network and Distributed Algorithm

We present shortly some definitions which are relevant to the paper, including graph theory and asynchronous algorithms. Let $\mathcal{G}_c = (\{1,\dots,N\}, E_c)$ denote a directed, static graph (digraph). The set $\{1,\dots,N\}$ are the nodes of the graph, corresponding to unique identifiers of the agents. The set $E_c \subset \{1,\dots,N\}^2$ denotes the set of edges connecting two nodes. A digraph is said to be *strongly connected* if, for every pair of nodes $(i,j) \in \{1,\dots,N\} \times \{1,\dots,N\}$, there exists a path of directed edges that goes from i to j. The maximum distance between any pair of agents (i,j) in the graph, is the *diameter* $\mathrm{diam}(\mathcal{G}_c)$ of the graph $\mathcal{G}_c$. We consider in particular time-dependent digraphs of the form $\mathcal{G}_c(t) = (V, E(t))$, $t \in \mathbb{R}_{\geq 0}$, where t represents the universal time. The graph $\mathcal{G}_c(t)$ models the communication in the network in the sense that at time t there is an edge from node i to node j if and only if agent i transmits information to agent j at time t. We impose the following assumption on the connectivity of the communication graph $\mathcal{G}_c$.

Assumption 2.1 (Periodically Strong Connectivity): There exists a positive and bounded constant T_c such that for every time instant $t \in \mathbb{R}_{\geq 0}$, the digraph $\mathcal{G}_c^{T_c}(t) := \cup_{\tau=t}^{T_c} \mathcal{G}_c(\tau)$ is strongly connected.

The agents in the network perform a *distributed algorithm* [9] to solve the optimization problem. In what follows, the superscript $[i]$ denotes that a quantity belongs to agent i. A distributed algorithm consists of: (1) the set W, called the set of *states* $w^{[i]}$, (2) the set Σ, called the *communication alphabet* including the null element, (3) the map $\mathtt{MSG} : W \times (1,\dots,N) \to \Sigma$, called *message function*, and (4) the map $\mathtt{STF} : W \times \Sigma^N \to W$, called the *state transition function*. We denote $t_k^{[i]}$ the time instants at which agent i updates its internal state. In this sense k is a counter for updates performed by an agent. Between two discrete updates, the state is constant $w^{[i]}(t) = w(t_k^{[i]})$ for all $t_k^{[i]} \leq t < t_{k+1}^{[i]}$. The evolution of the distributed algorithm is then as follows. The algorithm starts at $t = 0$ and each agent initializes its state to $w^{[i]}(0)$. Each agent performs two actions repeatedly: (i) the ith agent sends to each of its outgoing neighbors in the communication graph a message computed as $\mathtt{MSG}(w^{[i]}(t_k^{[i]}))$; (ii) whenever it receives information from its in-neighbors, it updates its state $w^{[i]}(t_{k+1}^{[i]})$ according to the state transition function. Each agent performs these two actions at its own speed and independent of the speed of the other agents. No synchronization is required in the network. Following [10], we say that the algorithm is *partially asynchronous* since it performs asynchronously, but T_c imposes a global bound on the time allowed to pass between consecutive state updates.

III. A Review on Column Generation

The problem OP has a very characteristic structure. Without the coupling constraint (2) the problem would fall into several independent problems. Each problem could then be solved independently of the others.

One of the best known techniques for handling those almost separable optimization problems is known as the "Dantzig-Wolfe Decomposition" [11], [12], [13]. The basic idea of the decomposition is that a large-scale linear program of a form OP can be decomposed into a coordinating master program MP and several smaller sub-problems SP_i. The original linear program can then be solved by alternately solving the master program and the sub-problems. We review the idea here.

A. Extensive Problem Representation

The method utilizes the fact that the polyhedron $\mathcal{X}_i$ can be uniquely represented as the polytope, formed by the convex combination of its extreme points, plus the polyhedral cone, formed by its extreme rays [14]. Let the polyhedron be generated by a finite number of half spaces, i.e. $\mathcal{X}_i = \{x_i \in \mathbb{R}^{n_i} | \mathrm{D}_i x_i \leq \mathrm{b}_i\}$. The *extreme points* x_i^p of the polyhedron are all points in $\mathcal{X}_i$ which cannot be represented as mid points of a line segment joining two other points of $\mathcal{X}_i$. The rays of the polyhedron x_i^r are the homogeneous solutions, i.e. $\mathrm{D}_i x_i^r = 0$. A ray x_i^r is an *extreme ray* if there are not two rays x_i^{r1} and x_i^{r2} ($x_i^{r2} \neq \alpha x_i^{r1}$) such that $x_i^r = \frac{1}{2}(x_i^{r_1} + x_i^{r_2})$. We define the index set of all extreme points $\mathcal{P}_I$ and extreme rays $\mathcal{R}_I$, respectively. We can now represent the polyhedron in an alternative form.

Theorem 3.1: [14] Let $\mathcal{X}_i$ be nonempty then it can be represented as

$$\mathcal{X}_i = \{\xi \in \mathbb{R}^{n_i} |\, \xi = \sum_{p\in\mathcal{P}_i} \lambda_{ip} x_i^p + \sum_{r\in\mathcal{R}_i} \lambda_{ir} x_i^r; \quad \sum \lambda_{ip} = 1,\ \lambda_{ip}, \lambda_{ir} \geq 0\},$$

where $\{x_i^p\}$ and $\{x_i^r\}$ are the extreme points and extreme rays of $\mathcal{X}_i$ respectively. □

This result allows to write any vector $x_i \in \mathcal{X}_i$ as

$$x_i = \sum_{p\in\mathcal{P}_i} x_i^p \lambda_{ip} + \sum_{r\in\mathcal{R}_i} x_i^r \lambda_{ir}, \ \sum_{p\in\mathcal{P}_i} \lambda_{ip} = 1, \tag{3}$$

where $x_i^{p,r} \in \mathbb{R}^{n_i}$ and $\lambda_{ip}, \lambda_{ir} \in \mathbb{R}$. We can combine this idea with the optimization problem. In particular the cost and the coupling constraints can be rewritten as

$$\mathrm{c}_i^T x_i = \sum_{p\in\mathcal{P}_i} \mathrm{c}_i^T x_i^p \lambda_{ip} + \sum_{r\in\mathcal{R}_i} \mathrm{c}_i^T x_i^r \lambda_{ir},$$

and

$$\mathrm{A}_i x_i = \sum_{p\in\mathcal{P}_i} A_i x_i^p \lambda_{ip} + \sum_{r\in\mathcal{R}_i} \mathrm{A}_i x_i^r \lambda_{ir}.$$

Defining $c_{ip} := \mathrm{c}_i^T x_i^p$ ($c_{ir} := \mathrm{c}_i^T x_i^r$) and $A_{ip} := \mathrm{A}_i x_I^p$ ($A_{ir} := \mathrm{A}_i x_i^r$), the cost of one agent can be represented as

$$\mathrm{c}_i^T x_i = \sum_{p\in\mathcal{P}_i} c_{ip}\lambda_{ip} + \sum_{r\in\mathcal{R}_i} c_{ir}\lambda_{ir}. \tag{4}$$

The coupling constraint can be similarly represented. Note that $c_{ip} \in \mathbb{R}$ ($A_{ip} \in \mathbb{R}^d$), while $\mathrm{c}_i \in \mathbb{R}^{n_i}$ ($\mathrm{A}_i \in \mathbb{R}^{d\times n_i}$).

Now we are ready to re-state the optimization problem *OP* in the *extensive form:*

$$\min \phi := \sum_{i=1}^{N} \{ \sum_{p\in\mathcal{P}_i} c_{ip}\lambda_{ip} + \sum_{r\in\mathcal{R}_i} c_{ir}\lambda_{ir}\} \tag{5}$$

$$\text{s.t.} \sum_{i=1}^{N} \{ \sum_{p\in\mathcal{P}_i} A_{ip}\lambda_{ip} + \sum_{r\in\mathcal{R}_i} A_{ir}\lambda_{ir}\} = \mathrm{b}_0 \tag{6}$$

$$\sum_{p\in\mathcal{P}_i} \lambda_{ip} = 1,\ \lambda_{ip} \geq 0, \lambda_{ir} \geq 0, \quad i \in \{1,\dots,N\} \tag{7}$$

We call this problem in extensive form the *master program* (MP). The master program is exactly equivalent to OP, in the sense that: (i) an optimal point in MP is an optimal point in OP; (ii) MP is unbounded if and only if OP is unbounded; and (iii) MP is infeasible if and only if OP is infeasible [13].

The new problem formulation does not necessarily seem to be favorable. The problem has been significantly blown up. While the original problem had $n = \sum_{i=1}^{N} n_i$ decision variables, the new problem has $n' = \sum_{i=1}^{N} (|\mathcal{P}_i| + |\mathcal{R}_i|)$ decision variables. Since the number of extreme points and extreme rays can be extremely

large, the master problem might have a huge number of decision variables. The benefit of the extensive representation lies in the gained structure of the problem. As commonly done in linear programming, we can group the problem information of MP into columns. A column of MP takes always the form [1]

$$\mathrm{h}_{i\kappa} = [c_{i\kappa}\ A_{i\kappa}^T\ e_{i\kappa}^T]^T. \tag{8}$$

Note that this representation is equivalent to $\mathrm{h}_{i\kappa} = [\mathrm{c}_i^T x_i^\kappa, (\mathrm{A}_i x_i^\kappa)^T, e_{i\kappa}]^T$ where x_i^κ is an extreme point or extreme ray, respectively. We say the column $\mathrm{h}_{i\kappa}$ is *generated* by $x_{i\kappa}$. The vector $e_{i\kappa} \in \mathbb{R}^N$ is an all zero vector, with only a single entry, $[e_{i\kappa}]_i = 1$, if the corresponding x_i^κ is an extreme point. Note that $\mathrm{h}_{i\kappa} \in \mathbb{R}^{1+d+N}$. Concerning the problem structure, we want to point out that the number of constraints in MP is significantly reduced compared to OP, leading to a (often favorable) problem with a large variables-to-constraints ratio. The Distributed Simplex algorithm [8] is best suited for problems of this structure. However, the drawback still remains in the huge number of columns. It is obviously not computationally effective to compute a-priori all the extreme points and extreme rays of all sets $\mathcal{X}_i$. To handle this problem, a *column generation* method is used to compute online new (and only relevant) columns. We review the idea here.

B. Column Generation

To streamline the notation, we will use in the following bold letters to denote the problem data of the master problem, e.g., $\boldsymbol{c}^T = [c_{11}, c_{12}, \ldots]$ and

$$\boldsymbol{A} = \begin{bmatrix} A_{11} & A_{12} & \ldots \\ e_{11} & e_{12} & \ldots \end{bmatrix}.$$

The matrix $\boldsymbol{A}$ includes now also the convexity constraints. The vector $\boldsymbol{c}$ and the matrix $\boldsymbol{A}$ have a large number of entries and columns, respectively. Therefore, we will never work with the complete vector or matrix, but only with subsets of them. Let $\mathbb{H}$ be a set of columns $\{\mathrm{h}_{ik}\}$ of MP, then $\boldsymbol{c}_H$ and $\boldsymbol{A}_H$ denote the subvector and submatrix, respectively, which correspond to the columns in $\mathbb{H}$. A *basis* of the master problem $\mathbb{B}$ is a set of $d+N$ columns $\{\mathrm{h}_{i\kappa}\}$ such that the corresponding constraint matrix $\boldsymbol{A}_B$ is invertible. A basis corresponds always to a feasible solution in the sense that a feasible vector of decision variables $\lambda_{i\kappa}$ can be computed from $\mathbb{B}$ with $\lambda_{i\kappa}$ nonzero only if $\mathrm{h}_{i\kappa} \in \mathbb{B}$.

Assume now, that a basis $\mathbb{B}$ for the problem MP is known. The *dual solution* corresponding to the basis $\mathbb{B}$ is

$$\begin{bmatrix} y \\ \pi \end{bmatrix} = (\mathbf{A}_B^{-1})^T \boldsymbol{c}_B,$$

where $y \in \mathbb{R}^d$ is the Lagrange multiplier which belongs to the coupling constraint (6) and $\pi = [\pi_1, \ldots, \pi_N]^T \in \mathbb{R}^N$ are the multipliers corresponding to the convexity constraints (7). Given a new, non-basic column $\mathrm{h}_{i\kappa} = [c_{i\kappa}, A_{i\kappa}^T, e_{i\kappa}^T]^T$, we ask now, when this column can improve the basis. Therefore, we consider the *reduced cost* of the column:

$$r_{i\kappa} = c_{i\kappa} - y^T A_{i\kappa} - \pi^T e_{i\kappa}. \tag{9}$$

A column $\mathrm{h}_{i\kappa}$ can improve the basis if the reduced cost is negative, $r_{i\kappa} < 0$.[2] Given the full problem information, this decision could be made directly. If, however, the problem is given in the extensive form (5), and the column data is not explicitly available, the reduced cost is used to construct a new column. Applying the inverse coordinate change and writing $r_{i\kappa}$ in the original x-coordinates gives

$$r_{i\kappa} = \mathrm{c}_i^T x_i^\kappa - y^T \mathrm{A}_i x_i^\kappa - \pi^T e_{i\kappa},$$

where the last term has the particular structure

$$\gamma_{i\kappa} := \pi^T e_{i\kappa} = \begin{cases} \pi_i & \text{if } x_i^\kappa \text{ is an extreme point,} \\ 0 & \text{if } x_i^\kappa \text{ is an extreme ray.} \end{cases}$$

This allows the conclusion that increasing the corresponding variable $\lambda_{i\kappa}$ in the master program, which is equivalent to including the column $\mathrm{h}_{i\kappa}$, generated by x_i^κ in the basis $\mathbb{B}$, decreases the overall cost ϕ if and only if

$$(\mathrm{c}_i^T - y^T \mathrm{A}_i) x_i^\kappa \leq \gamma_{i\kappa}.$$

The extreme points and rays are not explicitly known. We know, however, that the solution x_i^κ of the linear optimization problem

$$SP_i \quad z_i := \max_{x_i} (\mathrm{c}_i^T - y^T \mathrm{A}_i) x_i \text{ s.t } x_i \in \mathcal{X}_i. \tag{10}$$

is an extreme point if x_i^κ is finite. If SP_i is unbounded it is unbounded along an extreme ray [12], which we call in this case also x_i^κ. Note that SP_i is a linear program, that can be solved by several algorithms. Once we have obtained the optimal value z_i^κ and the optimal solution or extreme ray x_i^κ, respectively, we can decide whether the basis can be improved by including the column generated by x_i^κ. Including the column $\mathrm{h}_{i\kappa} = [\mathrm{c}_i^T x_i^\kappa, (\mathrm{A}_i x_i^\kappa)^T, e_{i\kappa}]^T$ in the basis improves the optimal value ϕ if and only if $z_i^\kappa < \gamma_{i\kappa}$. Note that $\gamma_{i\kappa}$ takes a characteristic structure: for an extreme point $\gamma_{i\kappa} = \pi_i$ and for and extreme ray $\gamma_{i\kappa} = 0$.

This idea allows to *generate* new columns for the master program MP by solving the linear sub-problems SP_i. The classical Dantzig Wolfe Decomposition would now proceed as follows. At first, a central coordination unit solves the master program MP for an initial set of columns and determines the corresponding multipliers $(y, \pi_1, \ldots, \pi_N)$; the multipliers are then transmitted to the sub-problems SP_i, which use it to generate columns. Then with the new column information the central unit solves again the master problem. This structure is illustrated in Figure 1.

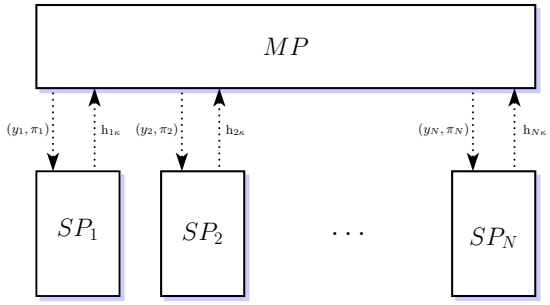

Fig. 1. Decomposition structure and information flow for the classical Dantzig-Wolfe decomposition.

In this work we take an alternative approach to the problem. We aim for an algorithm without any coordinating unit. We utilize the column generation idea to develop a Two-Stage Distributed Simplex algorithm for linear programs with local constraints.

IV. Two-Stage Distributed Simplex Algorithm

We consider now a multi-agent system with all the computation and communication restrictions described in § II-B. Our objective is to develop a distributed algorithm for solving the distributed linear program with local constraints OP. The new algorithm significantly generalizes our previous result in [8].

[1] To simplify the notation, we replace the indices p and r with one index κ. Whether κ refers to a particular extreme point of an extreme ray becomes clear from the context.

[2] It can be directly verified that, if $r_{i\kappa} < 0$, increasing the value of $\lambda_{i\kappa}$ from zero to a positive value decreases the overall cost ϕ.

A. Distributed Simplex

The Distributed Simplex algorithm is designed to solve distributed linear programs in multi-agent systems. It considers linear programs with only coupling constraints, and without the local constraints $x_i \in \mathcal{X}_i$ and is particularly well suited for problems with few constraints and many decision variables.

Before presenting the algorithm, we want to comment shortly on the problem of degeneracy in distributed linear programming. The issue is discussed more explicitly in [8]. Numerous linear programs happen to have more than one optimal solution. These problems are said to be *dual degenerate*. In distributed linear programming this becomes a critical issue which cannot be ignored. Indeed, any distributed algorithm has to ensure that all agents compute the *same* optimal solution. In [8] we proposed to use a lexicographic comparison for ensuring this. Lexicographic comparisons allow to define a unique ordering on vectors.

Definition 4.1 (Lex-positivity): If $v = (v_1, \dots, v_r)$ is a vector, then it is said to be *lex-positive* ($v \succ 0$) if $v \neq 0$ and the first non-zero component of v is positive. □

For two vectors, v and u, we say that $v \succ u$ if $v - u \succ 0$. The lexicographic minimum of a set of vectors $\{v_1, \dots, v_r\}$, denoted *lexmin*, is the element v_i, such that $v_j \succeq v_i$ for all $j \neq i$. For the same set of vectors, *lexsort*$\{v_1, \dots, v_r\}$ refers to the lexicographically sorted set of these vectors.

In [8] we used lexicographic ordering intensively to develop the Distributed Simplex algorithm, as a multi-agent version of the Simplex. Informally the Distributed Simplex algorithm can be described as follows. Each agent keeps a local version of the central problem in its memory, which is built from the limited information it has available. For this local problem, the agent computes a (locally optimal) basis $\mathbb{B}^{[i]}$, which represents also its internal state.

> *Distributed Simplex Algorithm*: Each agent consecutively performs the following tasks:
>
> (i) it transmits irregularly, but at least after a time interval of maximal length T_c, its basis to all its out-neighbors;
>
> (ii) whenever it acquires a basis from one of its in-neighbors, it sorts all columns in its memory according to a lexicographic ordering and
>
> (iii) it performs a `Simplex` algorithm with particular pivot rules to update its local basis.[3]

The algorithm uses the lexicographic ordering in the following sense. The computing agents perform two steps before updating the basis: (i) they sort the columns in their memory lexicographically (lexsort), (ii) they compute a lexicographically perturbed reduced cost. The reader is referred to [8] for details and the precise definition of the reduced cost. The relevant fact here is that this particular definition of the reduced cost ensures that all agents finally compute, out of all optimal bases, the unique basis which is lexicographically minimal.

Let us shortly discuss the problem data involved in step (ii) of the Distributed Simplex : (1) the current basis of the agent $\mathbb{B}^{[i]}$, (2) the columns received from the neighbors and (3) the *permanent columns* of the agent, i.e. the columns of the original problem which correspond to the decision variables of that particular agent. It is crucial for the correct convergence of the algorithm, that each agent considers its permanent columns in this step. The private information must be repeatedly evaluated. Based on this idea, we will now propose a new algorithm.

B. Two-Stage Distributed Simplex with Local Constraints

Consider now a problem of the form OP, including nontrivial local constraints $x_i \in \mathcal{X}_i$.

[3]Primal degeneracy, and therefore cycling of the algorithm is prevented by a classical perturbation.

Let us first point out that we could interpret OP as a distributed linear program in the sense of [8], if we consider the local constraints $x_i \in \mathcal{X}_i$ as global constraints. Then we could use the Distributed Simplex directly for solving this problem. However, the number of constraints forming the sets $\mathcal{X}_i$ might be extremely large and such a practice would require us to handle many coupling constraints. This would lead to extremely complex local computations and a significant overhead of data exchange between the agents.

We prefer to work in the following with the extensive problem formulation MP, instead of OP, since it has the preferred high columns-to-constraints ration. The large optimization problem MP is distributed to the individual agents and an algorithm inspired by the Distributed Simplex is proposed to solve it.

The idea of the new algorithm is as follows. Each agent $i \in \{1, \dots, N\}$ knows its own problem data $\mathrm{c}_i, \mathrm{A}_i, \mathcal{X}_i$ and the right-hand side of the coupling constraint b_0. The agent builds its own local version of the master program MP_i. The basis of this local master program is denoted by $\mathbb{B}^{[i]}$ and the corresponding value of the program by $\phi_B^{[i]}$. The basis $\mathbb{B}^{[i]}$ is the internal state of the agent i during the evolution of the algorithm. To initialize its local program MP_i, each agent creates artificial columns with a very high cost, to complete an initial basis (*big-M* method). The artificial columns will be eventually dropped during the evolution of the algorithm. Given a basis $\mathbb{B}^{[i]}$ for the master program MP_i the agent can compute the dual solution $(y^{[i]}, \pi^{[i]})$ and exactly define its subproblem SP_i. The algorithm evolves then according to the next rules, including the communication (C) and two internal stages of computation (S-I), (S-II).

> *Two-Stage Distributed Simplex:* Each agent repeats the following tasks:
>
> (C) it transmits irregularly, but at least after a time interval of maximal length T_c, its basis to all its out-neighbors;
>
> (S-I) whenever it acquires a basis from one of its in-neighbors, it sorts the received columns and its basis columns according to a lexicographic ordering and performs a `Simplex` algorithm with the particular pivot rules. It updates its local basis of MP_i and determines the dual variables $(y^{[i]}, \pi^{[i]})$ for this basis;
>
> (S-II) it generates a new column by solving the problem SP_i using the information $(y^{[i]}, \pi_i^{[i]})$ and eventually updates its basis.

The structure and the information flow of the algorithm are illustrated in Figure 2. We want to emphasize that the Two-Stage Distributed Simplex algorithm is a *fully distributed* algorithm without any coordination unit and requires only *partially asynchronous* communication.

The name "Two-Stage Distributed Simplex" emphasizes the structure of the algorithm. The internal computations performed by one agent deal with two optimization problems, the local master program MP_i and the sub-problem SP_i.

We have to adapt the algorithm only slightly to deal with an eventual unboundedness of the optimization problem OP. It is known [13] that the master program MP is unbounded exactly if the original program OP is unbounded. The same holds for the local master programs MP_i. If during the evolution of the algorithm one agent detects unboundedness, it sets it basis $\mathbb{B}^{[i]} = \texttt{null}$ and transmits this signal to all his neighbors. If an agent receives a `null` signal it also sets its basis to `null`. This allows the full network to detect unboundedness.

In the following table we provide a pseudo code of the algorithm.

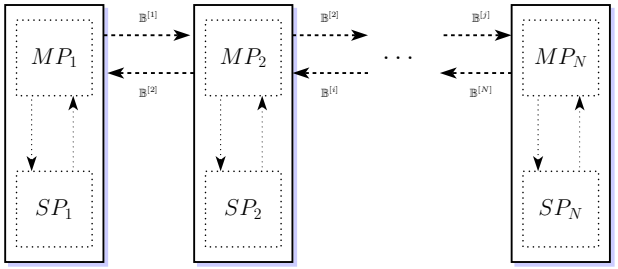

Fig. 2. Structure and information flow for the novel Two-Stage Distributed Simplex algorithm.

Problem data: $MP, \mathcal{G}_c$

Algorithm: Two-Stage Distributed Simplex

Message alphabet: $\Sigma = \{\mathrm{h}_{i\kappa}\} \cup \{\mathtt{null}\}$

Processor state: $\mathbb{B}^{[i]}$ *% basis of* MP_i

Initialization: $\mathbb{B}^{[i]} := \mathbb{B}_M$ *% big-M method*

function $\mathtt{MSG}(\mathbb{B}^{[i]}, j)$
  return all $\mathrm{h}^{[i]}$ contained in $\mathbb{B}^{[i]}$ but not in $\mathbb{B}_M$.

function $\mathtt{STF}(\mathbb{B}^{[i]}, y)$
% executed by agent i, with $y_j := \mathtt{MSG}(\mathbb{B}^{[j]}, i) = \mathbb{B}^{[j]}$
  if $y_j \neq \mathtt{null}$ for all $j \in \mathcal{N}_I(i)$ **then**
    % STAGE I
    $\mathbb{H}^{tmp} \leftarrow \mathrm{lexsort}\{\mathbb{B}^{[i]} \cup \big(\cup_{j\in\mathcal{N}_I(i)}\, y_j\big)\}$
    $(\mathbb{B}^{[i]}, y^{[i]}, \pi^{[i]}) \leftarrow \mathtt{Simplex}\ (\mathbb{H}^{tmp}, \mathbb{B}^{[i]})$ *% solve* MP_i
    % STAGE II
    $\mathbb{B}^{[i]} \leftarrow \mathrm{solve}\ SP_i(y^{[i]}, \pi_i^{[i]})$ *% generate new column*
  else
    $\mathbb{B}^{[i]} \leftarrow \mathtt{null}$ *% program is unbounded*
  end if

We want to comment shortly on communication and information storage requirements of the algorithm.

Remark 4.2 (Size of exchanged messages): The agents exchange the bases $\mathbb{B}^{[i]}$ of their local master programs MP_i. At each time instant, this basis $\mathbb{B}^{[i]}$ is the optimal basis with respect to all the information available to the agents. A basis consists of several columns $\mathrm{h}_{i\kappa}$. The following information needs to be transmitted to characterize $\mathrm{h}_{i\kappa}$ exactly: the problem data $c_{i\kappa}, A_{i\kappa}$, the index of the corresponding agent i, and a marker indicating whether the column is generated by an extreme point or an extreme ray. A column requires therefore a vector of $d+1$ (eventually) rational numbers, one identifier for an agent and one additional bit. □

Remark 4.3 (Private processing of local constraints): For a *reconstruction* of the original primal solution x_i^*, each agent must store the extreme points or extreme rays of the columns it generates. However, for each agent it suffices to know this information for its own columns, since it only needs to reconstruct its part of the optimal solution x_i^* (and not the complete vector x^*). Therefore, this information can remain private and needs *not* to be exchanged with other agents at any time. □

We can now prove that our new algorithm solves the problem OP correctly.

Theorem 4.4 (Two-stage Distributed Simplex analysis): Consider the distributed linear program OP in (1) and a multi-agent system with communication network as described in Section II-B running the *Two-Stage Distributed Simplex* algorithm. Then, there exists a finite time instant T_f at which, all agents,

(i) agree on a common optimal solution, if OP has a finite optimal solution;
(ii) detect unboundedness, if OP is unbounded;
(iii) detect infeasibility, if OP is infeasible. □

Proof: To prove the statement (i), we assume that the problem OP has a finite optimal solution. We show first that every agent will eventually determine an optimal solution (which could be distinct for different agents). Subsequently, we show, using a contradiction argument, that all agents must converge to the same optimal basis.

Consider the cost of the master program of each agent MP_i at the time instant t, $\phi_B^{[i]}(t)$.[4] All agents initialize their problems such that $\phi_B^{[i]}(0) < \infty$. Let ϕ^* be the minimal value of the master program MP. A fundamental result in linear programming says that from any basis, there is a sequence of pivot iterations to an optimal basis [12]. Thus, as long as $\phi_B^{[i]}(t) > \phi^*$, there exists at least one column h such that $\phi_{\{B\cup\mathrm{h}\}} < \phi_B$. The algorithm offers two possibilities how such a column h can be included into $\mathbb{B}^{[i]}$.

First, such a column h can be generated by the local subproblem SP_i. When for $\mathbb{B}^{[i]}(t)$ and the corresponding multipliers $(y^{[i]}(t), \pi_i^{[i]}(t))$ the optimal value $z_i^{[i]}(t) < \gamma^{[i]}(t)$, then $\phi_{B\cup\mathrm{h}} < \phi_B$, where h is the column generated by the optimal solution $x_i^{[i]}(t)$ of SP_i. In this case, the basis will improve after a finite time.

Second, if $z_i^{[i]} \geq \gamma^{[i]}(t)$, then there must be at least one agent j such that $\phi_B^{[i]}(t) > \phi_B^{[j]}(t)$. If this is true, there must be two (eventually other) agents k and l, with $\phi_B^{[k]} > \phi_B^{[l]}$ in the network, such that $(l,k) \in E_c(\tau), \tau \in \{t, t+T_c\}$. By the structure of the algorithm it follows that $\phi_B^{[k]}(\tau) \leq \phi_B^{[l]}(t) < \phi_B^{[k]}(t)$. We can now use the argumentation presented in [8], or similarly in [6], to show that under the connectivity assumption 2.1 of the communication network $\mathcal{G}_c$, there exists a time T such that $\phi_B^{[i]}(t+T) \leq \phi_B^{[j]}(t) < \phi_B^{[i]}(t)$.

In any case, as long as $\phi_B^{[i]}(t) > \phi^*$ there exists a finite time T_D such that $\phi_B^{[i]}(t+T_D) < \phi_B^{[i]}(t)$. Since there are only finitely many bases (and we are using a particular pivot rules to avoid cycling) there exists T_f' such that $\phi_B^{[i]}(T_f') = \phi^*$ for all $i \in \{1,\ldots,N\}$. At this point, any agent knows an optimal solution to the master program. However, it is still possible that $\mathbb{B}^{[i]}(T_f') \neq \mathbb{B}^{[j]}(T_f')$ for some agents i and j in the network.

We show now that there is finite time T_f at which the algorithm is converged and $\mathbb{B}^{[i]}(T_f) = \mathbb{B}^{[j]}(T_f)$ for all i, j. Convergence of the algorithm implies that $\mathbb{B}^{[i]}(T_f) = \mathbb{B}^{[i]}(T_f + \tau),\ \tau \geq 0$, for all agents $i \in \{1,\ldots,N\}$. We note that at the time all agents have an optimal basis no further columns will be generated by the sub-problems and, therefore, only the communication between the agents will be relevant.

Assume, in order to get a contradiction, that the algorithm has converged and there are two agents i and j such that $\mathbb{B}^{[i]}(T_f) \neq \mathbb{B}^{[j]}(T_f)$. We can assume without loss of generality that the basis $\mathbb{B}^{[j]}(T_f)$ has a *lexicographically* smaller cost than $\mathbb{B}^{[i]}(T_f)$ (see [8] for the definition), and that agent j directly communicates its basis to agent i. If agent i receives $\mathbb{B}^{[j]}(T_f)$ from agent j, it computes the *unique* lexicographically minimal (and optimal) basis contained in the union of the two bases, i.e., it preforms the two actions $\mathbb{H} = \mathrm{lexsort}\{\mathbb{B}^{[i]}(T_f) \cup \mathbb{B}^{[j]}(T_f)\}$, and $\mathtt{Simplex}\ (\mathbb{H}, \mathbb{B}^{[i]}(T_f)$. The new basis, computed after these two actions, must be lexicographically better that $\mathbb{B}^{[i]}(T_f)$ (since already $\mathbb{B}^{[j]}(T_f)$ was lexicographically better). Therefore the new basis cannot be identical to $\mathbb{B}^{[i]}(T_f)$. This contradicts the assumption that the algorithm has converged. We note that, with the same argumentation used previously, each agent will have a lexicographically better basis within a finite time interval. Since there is a sequence of pivot iterations, leading from any basis to another basis, we conclude that the algorithm will converge. Since there are only finitely many different bases, and no agent can recompute a worse basis, the algorithm will converge in

[4] Note that *no* lexicographically perturbed cost is considered here.

finite time. Summarizing, We conclude that the algorithm converges in finite time such that $\phi_B^{[i]} = \phi_B^{[j]} = \phi^*$ and $\mathbb{B}^{[i]} = \mathbb{B}^{[j]}$ for any two agents i, j.

The proof of the remaining two statements is analogous to the proof presented in [8] and is therefore omitted here.

■

To conclude the presentation of the algorithm we point out that each agent can halt the algorithm execution if the value of the basis has not changed in a time interval of length $(2\,\mathrm{diam}(\mathcal{G}_c) + 1)T_c$ [8], and knows therefore in a finite time T_f that it has achieved an agreement with the other agents on an optimal solution.

It is not obvious a-priori on which optimal basis the agents will agree. Assume that there are several optimal solutions. We focus now on the situation where all agents know a optimal solution $\phi_B^{[i]} = \phi_B^{[j]} = \phi^*$ but have different bases $\mathbb{B}^{[i]} \neq \mathbb{B}^{[j]}$. Each agent updates now its basis, based on the column information of its own basis and the bases received from neighbors. It keeps the resulting basis in its memory and drops all other columns. Linear programs do not satisfy the persistency property [6], and therefore columns might be lost at this point which would be part of the lexicographically minimal basis. If, however, such a column is only known by a single agent it will never be retained by the algorithm. The basis on which the agreement is achieved can then not contain this column any more. It depends on various parameters, like e.g. the communication network $\mathcal{G}_c(t)$, which columns will be lost during the evolution of the algorithm and therefore on which solution an agreement will be achieved. However, it suffices here to conclude that the agreement basis is an optimal basis. Therefore the algorithm solves the distributed linear program OP.

C. Discussion

We want to emphasize again that the main difference between the new Two-Stage Distributed Simplex and our previous Distributed Simplex algorithm lies in the way the private or local information is handled. Instead of storing the private information as columns (previously called "permanent columns"), it is now encoded in the sub-problems SP_i. The private information is thus no longer evaluated equally with all other information, in particular the information received from other agents. Now the smaller sub-problem SP_i is solved to generate a new column from the local information. In the column generation step *no* lexicographic ordering or comparison is required. Purely the standard problem SP_i is solved, which can be done by standard algorithms. As a consequence the complexity of the local problem, namely the number of columns on which the lexicographically perturbed simplex has to be performed, is reduced. It is still important for the algorithm's convergence, that the private information is repeatedly examined, but it has an influence only until an optimal basis is known and not until the algorithm has fully converged. If the problem OP is dual-degenerate and is solved with the two algorithms Two-Stage Distributed Simplex and the Distributed Simplex (which is ineffective but possible), the final bases which are computed by the algorithms might be different, although both are optimal.

Our new algorithm might seem to be close to the Dantzig-Wolfe decomposition. In fact, it utilizes the same column generation idea. The important distinction lies in the communication structure of the algorithms. In the classical Dantzig-Wolfe decomposition the subproblems are solved independently, after receiving information from a *central* coordination unit, which solves the master program. The master problem transmits *dual* variables to the subproblems and receives *primal* information (i.e., generated columns).

Our approach is clearly distinct from this. We do neither require a central master problem nor a central coordination unit. In fact, the former master problem is distributed to the agents, which all keep their individual local copies. The local computations within one agent are such that alternating the local master program and the local sub-problem are solved (therefore it is a two-stage algorithm). The information exchange between the distributed decision makers is now only restricted to *primal* information. No dual variables are exchanged between the agents. This simple communication structure together with the only nearest-neighbor communication requirements and the inherent asynchronous structure, make our algorithm perfectly suited for multi-agent systems, where communication structures are unknown a-priori or change over time.

V. Conclusions

We propose a novel distributed algorithm to solve distributed linear programs with local constraints within asynchronous multi-agent systems. We propose a two-stage method, which encodes private information of an agent in a sub-problem, which is repeatedly solved in a local manner. The new algorithm allows to handle problems with a large number of local constraints, if only the number of coupling constraints is limited. One can interpret the new algorithm as a generalization of our previously proposed Distributed Simplex. It is an optimization algorithm which does not require any coordination unit and which can perform on asynchronous communication networks. Instead of alternately communicating primal and dual information, we only exchange primal information between the decision makers. We prove that, even if multiple optimal solutions exist, all agents agree on a common optimal solution in finite time, if one exists. On which optimal solution the agreement will be achieved cannot easily be predicted a priori, but depends on the evolution of the algorithm.

References

[1] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[2] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[3] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints via primal-dual subgradient methods," *IEEE Transactions on Automatic Control*, 2011, to appear.

[4] G. Yarmish and R. Slyke, "A distributed, scalable simplex method," *Journal of Supercomputing*, vol. 49, pp. 373–381, 2009.

[5] H. Dutta and H. Kargupta, "Distributed linear programming and resource management for data mining in distributed environments," in *IEEE International Conference on Data Mining Workshops*, 2008, pp. 543–552.

[6] G. Notarstefano and F. Bullo, "Distributed abstract optimization via constraints consensus: Theory and applications," *IEEE Transactions on Automatic Control*, 2011, accepted. [Online]. Available: http://motion.me.ucsb.edu/pdf/2007z-nb.pdf

[7] ——, "Network abstract linear programming with application to minimum-time formation control," in *IEEE Conference on Decision and Control*, New Orleans, LA, December 2007, pp. 927–932.

[8] M. Bürger, G. Notarstefano, F. Allgöwer, and F. Bullo, "A distributed simplex algorithm and the multi-agent assignment problem," in *Proc. of American Control Conference*, San Francisco, USA, July 2011, pp. 2639 – 2644.

[9] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009. [Online]. Available: http://coordinationbook.info

[10] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computations: Numerical Methods*. Belmont, Massachusetts: Athena Scientific, 1997.

[11] G. Dantzig and P. Wolfe, "The decomposition algorithm for linear programs," *Econometrica*, vol. 29, no. 4, pp. 767 – 778, 1961.

[12] G. B. Dantzig, *Linear Programming and Extensions*. Princeton University Press, 1963.

[13] J. R. Tebboth, "A computational study of dantzig-wolfe decomposition," Ph.D. dissertation, University of Buckingham, 2001.

[14] A. Schrijver, *Theory of Linear and Integer Programming*. Wiley, 1986.